International Journal of Engineering Applied Sciences and Technology, 2021
Vol. 5, Issue 10, ISSN No. 2455-2143, Pages 233-237
Published Online February 2021 in IJEAST (http://www.ijeast.com)

YOLO OBJECT DETECTION USING OPENCV

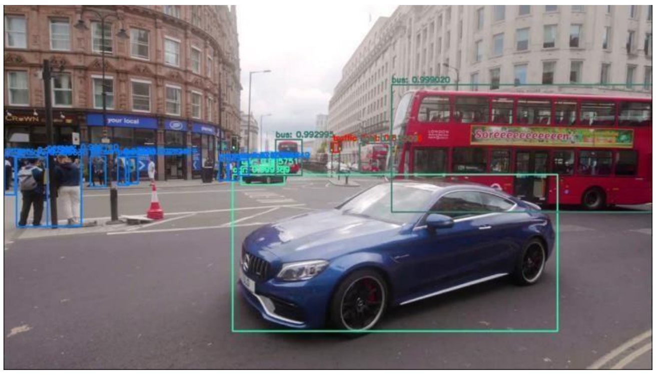
Akshara Gupta
BCA-IOP (SCSE)
Galgotias University, Kanpur, U.P., India

Aditya Verma
BCA-IOP (SCSE)
Galgotias University, Noida, U.P., India

Aditya Yadav
BCA-IOP (SCSE)
Galgotias University, Kanpur, U.P., India

Mr. Arvidhan M
Associate Professor (SCSE)
Galgotias University, Noida, U.P., India

Abstract— **The Object detection has significant problems with computer intelligence and vision in which we create algorithms to recognize objects where and where the object is located inside when given input (image) where there is more input. The correct problem of of object detection is very difficult, and can also see objects but does do not say where the required object is extracted.**



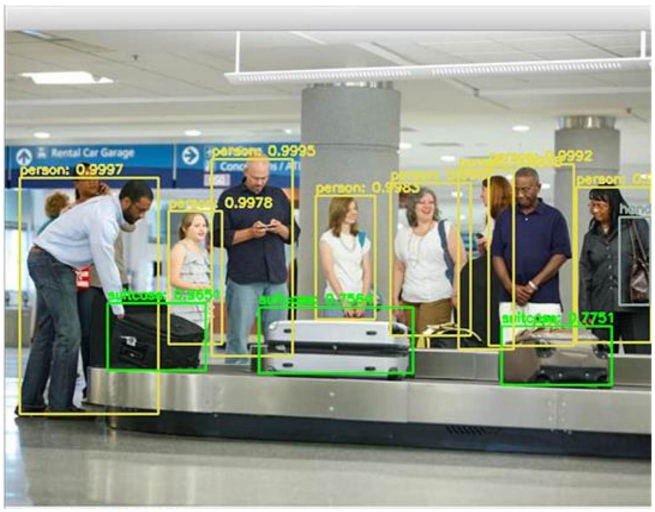
Keywords— **YOLO Object, OpenCV, Object.**

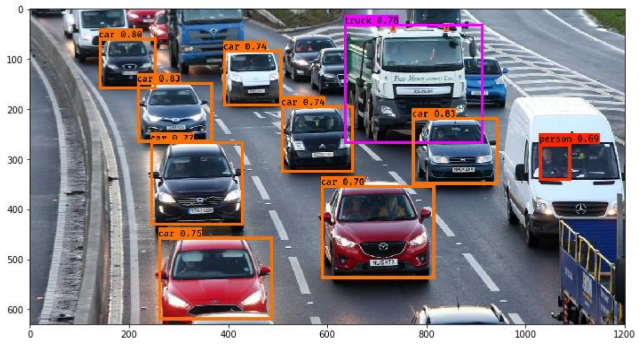
I. INTRODUCTION

The accuracy of the Object detection is a technology that protects variations of the same objects of the same type (category) of computer and videos. Foremost common object to detect on time this application is a variety of vehicles and other means of transportation and footing. If we want to exclude different objects from an event we have to use a certain process as Object Localization and has to locate one object in the redesigned time plans.

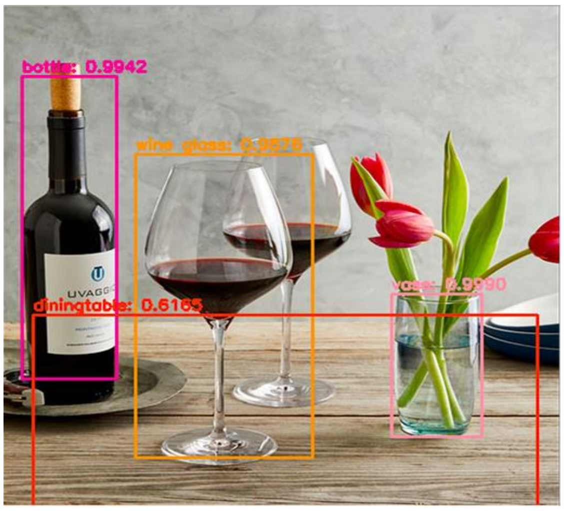
Generally in a real-time algorithm, it takes 45 frames to processes per second whereas in YOLO it makes fewer errors and also predicts less false positives within the input.

Example Predictions made by YOLO





```
Found 10 boxes for img.jpg
person 0.69 (1027, 270) (1091, 343)
car 0.70 (587, 352) (817, 539)
car 0.74 (390, 82) (509, 194)
car 0.74 (507, 193) (653, 327)
car 0.75 (259, 455) (490, 621)
car 0.77 (243, 265) (431, 432)
truck 0.78 (634, 30) (915, 269)
car 0.80 (139, 58) (255, 160)
car 0.83 (216, 146) (369, 267)
car 0.83 (771, 216) (945, 351)
```



II. MODULES

- **Training Dataset**

One of the most important parts of a machine learning program is to collect high- quality data. You can expect to spend more time on data. Important because our model is the data you only learn from data you only learn from. So how do we teach a machine to get hats? As you can probably guess, by showing many examples.

There is a various ways to collect data. When it comes to minges, any of the most popular items use Google Image Search.

- **Label data**

Once the time is up, the next step is to explain/explain it. In the future of discovery, labeling means drawing boxes and objects we wish to find them in pictures and give them a chance to find the corresponding categories / categories of objects so that we need to enjoy the pictures one by looking at high-quality things in hand.

- **Object Discovery**

Camera-yolo3 project provides a wide range of capa abilities to use YOLOv3 modes, including object acquisition, transfer learning, and training of new models from cs sc. In this set, we will use model to perform object detection in an invisible view. This functionality is found in a single Python file in a repository named "yolo3_one_file_to_detect_them_all.py" with 435 approximately cables. This text, in effect, will use pre-trained tools to prepare a model and then use model to perform object detection and output modeling. Released on OpenCv. To apply using this program aggressively, we will use the drawings available in this program and imrrove our documentation to start preparing and maintaining the Keras YOLOv3 model and bese download the model to make a new image prediction.

Technical Performance:

Program will use Python and OpenCv to train and reconfigured using YOLO Dataset which freely defined online. Program does require Computer System and GPU to train and perform Object Detection. Program also requires Webcam to re-evaluate objects during re-use of webcam.

- **Economic Access**

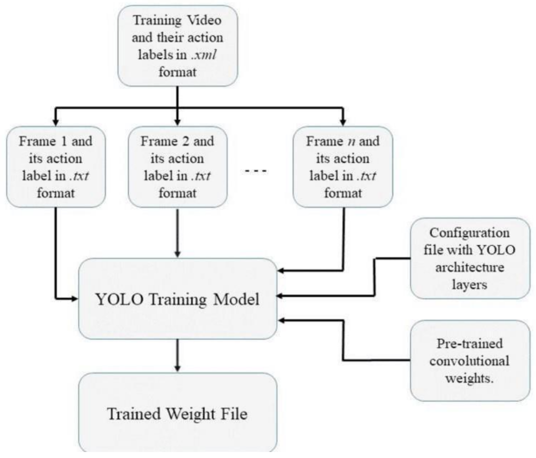
Proposed Program requires development tools and software such as PyCharm and KOLO data sets which
free of charge online.

For developing the proposed program, we need a various resources such as computer systems, internet connection for help, recommended disk space, and memory speed as mention to the needs of technical. By acquiring all of these features and releasing the proposed app, we have a huge advantage.

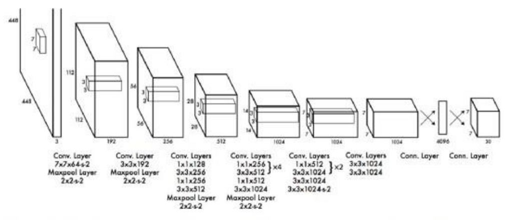
III. PROPOSED PROGRAM SUCH ARE

- The existing anp does not use the new YOLO Object Versions and algorithms, wheretechnology may not be accurate, up to date, and most of the time. However, the proposed system will be deleted, so we can only recover from the existing system.

- Also with this program new insertion, Therefore, and modification of various data will be very difficult. This program will shock you paperwork. Quality of the dataset will be written So keeping all the above-mentioned benefits and comparing with various expenditures of resources, we conclude that the proposed is economically feasible.


Training Video and their action labels in .xml format
Frame 1 and its action label in .txt format
Frame 2 and its action label in .txt format
- - -
Frame n and its action label in .txt format
Configuration file with YOLO architecture layers
YOLO Training Model
Pre-trained convolutional weights.
Trained Weight File

IV. HOW DOES YOLO WORK

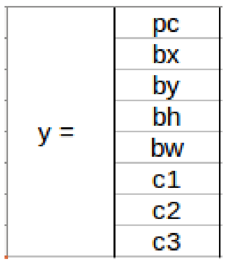

448
448
112
112
56
56
28
28
14
14
7
7
3
192
256
512
1024
1024
1024
4096
30
Conv. Layer 7x7x64-s-2 Maxpool Layer 2x2-s-2
Conv. Layer 3x3x192 Maxpool Layer 2x2-s-2
Conv. Layers 1x1x128 3x3x256 1x1x256 3x3x512 Maxpool Layer 2x2-s-2
Conv. Layers 1x1x256 3x3x512 }x4 1x1x512 3x3x1024 Maxpool Layer 2x2-s-2
Conv. Layers 1x1x512 3x3x1024 }x2 3x3x1024 3x3x1024-s-2
Conv. Layers 3x3x1024 3x3x1024
Conn. Layer
Conn. Layer

- YOLO first takes an information picture:

- The framework then partitions the input image into frameworks (say a 3 X 3 network):

Image classification and localization are applied on each matrix. YOLO then predicts the bounding boxes and their corresponding class probabilities for objects (if any are found, of course).

Suppose we have partitioned the image into a framework of size 3 X 3 and there are a total of 3 classes in which we want the objects to be classified. We should say the classes are Pedestrian, Car, and Motorcycle respectively. So, for each framework cell, the label y will be an eight-dimensional vector:

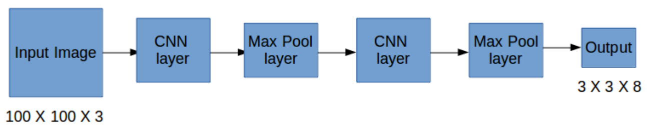


y =	pc
	bx
	by
	bh
	bw
	c1
	c2
	c3

- pc characterizes whether an object is present in the lattice or not (it is the probability) .

- bx, by, bh, bw specify the bounding box if there is an object .

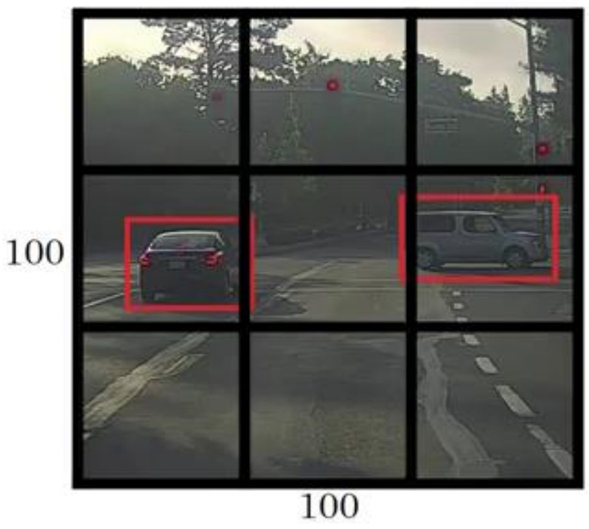
- c1, c2, c3 represent the classes. So, if the object is a car, c2 will be 1 and c1 and c3 will be 0, and so on.

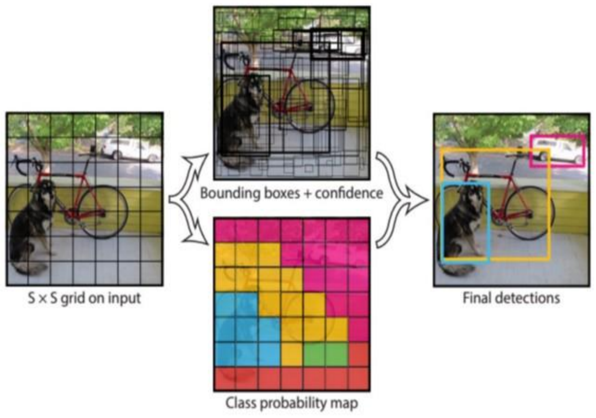


V. CONFIDENCE SCORE

- Yolo uses confidence score to identify the object and give them score which determine the probability of the prediction to be true by comparing.
- The algorithm is given below for the confidence score :

-
  ```
  # initialize our lists of detected bounding boxes, confidences, and
  # class IDs, respectively
  boxes = []
  confidences = []
  classIDs = []
  ```



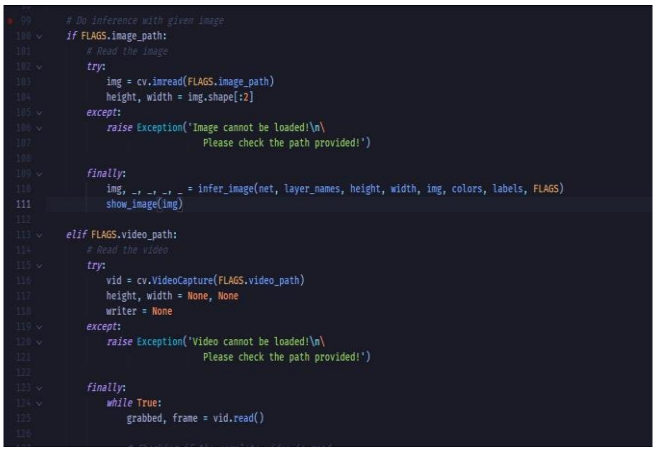


- Each grid cell predicts bounding boxes and confidence scores for those boxes. These confidence scores reflect how confident the model is that the box contains an object and also how accurate it thinks the box is that it predicts.

- If no object exists in a cell, its confidence score should be zero.

VI. EXCEPTIONS

- Yolo can also handle image exceptions and mark them as flags or exceptions.



```
    # Do inference with given image
    if FLAGS.image_path:
        # Read the image
        try:
            img = cv.imread(FLAGS.image_path)
            height, width = img.shape[:2]
        except:
            raise Exception('Image cannot be loaded!\n\
                            Please check the path provided!')

        finally:
            img, _, _, _, _ = infer_image(net, layer_names, height, width, img, colors, labels, FLAGS)
            show_image(img)

    elif FLAGS.video_path:
        # Read the video
        try:
            vid = cv.VideoCapture(FLAGS.video_path)
            height, width = None, None
            writer = None
        except:
            raise Exception('Video cannot be loaded!\n\
                            Please check the path provided!')

        finally:
            while True:
                grabbed, frame = vid.read()
```

VII. LIMITATIONS OF YOLO

- YOLO imposes strong spatial constraints on bounding box predictions since each matrix cell only predicts two boxes and can only have one class.

- This spatial constraint limits the number of nearby objects that our model can predict. Our model battles with small objects that appear in groups, such as flocks of winged animals.

- Since our model learns to predict bounding boxes from data, it battles to generalize to objects in new or unusual aspect ratios or configurations.

- our model also utilizes relatively coarse features for predicting bounding boxes since our architecture has multiple downsampling layers from the input image.

- Finally, while we train on a loss function that approximates detection performance, our loss function treats errors the same in small bounding boxes versus large bounding boxes. A small error in a large box is generally generous yet a small error in a small box has a much greater effect on IOU.

- The main source of error is incorrect localizations.

VIII. CONCLUSION

The accuracy of the Object detection in the real world is great and very powerful for many computers and Artificial vision systems. Great progress has been around for the past few years, and the current methods of used equipment have been linked to illegal driving, we are still not happy with the high performance, new driving methods. It should be used instead of being used where it can provide many benefits to the general public.
Robotics, and operating in artificial intelligence technology, the discovery that they can be widely used as drones and other UAVs where detection technology is becoming increasingly common.

IX. REFERENCES

1. Alvarez-Jimenez, Charkes Daikin. (2013). Why does human vision be superior to any other mammal, 143(1), 143– 149.

2. Anwas Jiwabi., Stip, E., & Kara, N. (2017).Learning Computer Vision through Magic and Python., 12(1), 70– 76.

3. Georgia Razdana, Mukhtar Abbas, Amit Sharma (2018), Classifying everyday objects through computer vision, (13)(1)

4. Ashley Main, Tom Scott, YOLO major advantages of a real computer vision system that generates reliable output.

5. Gaurav Dadhich, Shruti Sambit, Classifying objects based on their outline and greyscale image (2018) (7).

6. D. Erhan, C. Szegedy, A. Toshev, and D. Anguelov. Scalable object detection using deep neural networks. In Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on, pages 2155–2162. IEEE, 2014. 5, 6

7. M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes challenge: A retrospective. International Journal of Computer Vision, 111(1):98–136, Jan. 2015. 2

8. P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. IEEE Transactions on Pattern Analysis and Machine Intelligence, 32(9):1627–1645, 2010. 1, 4

9. S. Gidaris and N. Komodakis. Object detection via a multiregion & semantic segmentation-aware CNN model. CoRR, abs/1505.01749, 2015. 7

10. S. Ginosar, D. Haas, T. Brown, and J. Malik. Detecting people in cubist art. In Computer Vision-ECCV 2014 Workshops, pages 101–116. Springer, 2014.

11. R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on, pages 580–587. IEEE, 2014. 1, 4, 7 .

12. R. B. Girshick. Fast R-CNN. CoRR, abs/1504.08083, 2015. 2, 5, 6, 7.

13. Prof. Jaima Nelson, Kapur Naman (2017).Object Detection using CNN and Deep CNN., 11(1), 72–76.

14. Anderson Meo, Mustafa Ahmed(2019). Differentiating Ordinary Objects through Computer V, 37, 72–112.